IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Boucher, et al.	Serial No.:	Unknown
Filed:	June 19, 2003	Examiner:	Unknown
Docket No.	ALA-008F	GAU:	Unknown
Assignee:	Alacritech Inc.		
Title:	HIGH PERFORMANCE NETWORK INTERFACE		

Mail Stop PATENT APPLICATION June 19, 2003
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST TO PROVOKE INTERFERENCE

Sir:

Applicants hereby seek to have an interference declared between the present application and U.S. Patent No. 6,453,360, issued September 17, 2002.

U.S. Patent No. 6,453,360 to Muller et al. issued from U.S. application serial number 09/259,765, filed March 1, 1999.

The present application claims the benefit under 35 U.S.C. §120 of (is a continuation of) U.S. Patent Application Serial No. 10/005,536, filed November 7, 2001, which in turn claims the benefit under 35 U.S.C. §120 of (is a continuation of) U.S. patent application serial number 09/384,792, filed August 27, 1999, now U.S. Patent No. 6,434,620, which in turn: 1) claims the benefit under 35 U.S.C. §119 of provisional patent application serial number 60/098,296, filed August 27, 1998, 2) claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. patent application serial number 09/067,544, filed April 27, 1998, now U.S. Patent No. 6,226,680, and 3) claims the benefit under 35 U.S.C.

§120 of (is a continuation-in-part of) U.S. patent application serial number 09/141,713, filed August 28, 1998, now U.S. Patent No. 6,389,479.

U.S. Patent No. 6,226,680 and U.S. Patent No. 6,389,479 both claim the benefit under 35 U.S.C. §119 of provisional patent application serial number 60/061,809, filed October 14, 1997. The present application also claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. Patent Application Serial No. 09/464,283, filed December 15, 1999, now U.S. Patent No. 6,427,173, and claims the benefit under 35 U.S.C. §120 of (is a continuation-in-part of) U.S. Patent Application Serial No. 09/514,425, filed February 28, 2000, now U.S. Patent No. 6,427,171.

Claims 1-23, 27-28, 31-40, 42-45, 47-56 and 58-62 of the present application are copied from Claims 1-23, 27-28, 31-40, 42-45, 47-56 and 58-62 of U.S. Patent No. 6,453,360, respectively.

REQUIREMENTS OF 37 CFR 607:

Per rule 607(a)(1), the patent is U.S. Patent No. 6,453,360.

Per rule 607(a)(2), Applicants submit Counts 1-54 as set forth below.

Count #1 (corresponds exactly to Claim 1 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #2 (corresponds to Claim 2 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol, wherein said parsing comprises:

copying a header portion of said first packet into a header memory; and

examining said header portion according to a series of parsing instructions;

wherein said parsing instructions are configured to reflect a set of pre-selected communication protocols;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #3 (corresponds to Claim 3 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol, wherein said parsing comprises:

copying a header portion of said first packet into a header memory; and

examining said header portion according to a series of parsing instructions;

wherein said parsing instructions are configured to reflect a set of pre-selected communication protocols and said parsing instructions are updateable;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #4 (corresponds to Claim 4 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol, wherein said parsing comprises:

copying a header portion of said first packet into a header memory; and

examining said header portion according to a series of parsing instructions;

wherein said parsing instructions are configured to reflect a set of pre-selected communication protocols;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

copying a value from a field in a header of said header portion.

Count #5 (corresponds to Claim 5 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol, wherein said parsing comprises:

extracting an identifier of a source of said first packet from said header portion; and

extracting an identifier of a destination of said first packet from said header portion;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #6 (corresponds to Claim 6 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol, wherein said parsing comprises:

extracting an identifier of a source of said first packet from said header portion; and

extracting an identifier of a destination of said first packet from said header portion;

generating a flow key to identify a first communication flow that includes said first packet, wherein said generating comprises combining said source identifier and said destination identifier;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #7 (corresponds to Claim 7 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet, wherein said generating comprises retrieving an identifier of a communication connection from said header portion;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #8 (corresponds to Claim 8 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

storing said first packet in a packet memory prior to said transferring.

Count #9 (corresponds to Claim 9 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

storing said flow key in a flow database, wherein said flow database is configured to facilitate management of said first communication flow.

Count #10 (corresponds to Claim 10 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

storing said flow key in a flow database, wherein said flow database is configured to facilitate management of said first communication flow; and associating a flow number with said first packet, wherein said flow number comprises an index of said flow key within said flow database.

Count #11 (corresponds to Claim 11 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

storing said flow key in a flow database, wherein said flow database is configured to facilitate management of said first communication flow;

associating a flow number with said first packet, wherein said flow number comprises an index of said flow key within said flow database; and

storing said flow number in a flow memory.

Count #12 (corresponds to Claim 12 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

storing said flow key in a flow database, wherein said flow database is configured to facilitate management of said first communication flow; and

updating an entry in said flow database associated with said flow key when a second packet in said first communication flow is received.

Count #13 (corresponds exactly to Claim 13 of the '360 patent):

A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet received at a network interface from a network to a host computer system, the method comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol, storing

said packet in a non-re-assembly buffer.

Count #14 (corresponds to Claim 14 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, wherein said associating comprises:

retrieving one or more header fields of said header portion; and

analyzing said header fields to determine said status of said first packet.

Count #15 (corresponds to Claim 15 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, wherein said associating comprises:

retrieving one or more header fields of said header portion; and

analyzing said header fields to determine said status of said first packet, wherein said analyzing comprises:

determining whether said first packet includes a data portion; and

if said first packet includes a data portion, determining whether said data portion exceeds a pre-determined size.

Count #16 (corresponds to Claim 16 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet, wherein said associating comprises:

retrieving one or more header fields of said header portion; and

analyzing said header fields to determine said status of said first packet, wherein said analyzing comprises determining whether said first packet was received out of order in said first communication flow.

Count #17 (corresponds to Claim 17 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

storing said operation code in a control memory.

Count #18 (corresponds to Claim 18 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

wherein said first packet is determined to conform to said pre-selected protocol, said transferring comprising:

storing a data portion of said first packet in a re-assembly storage area, wherein said re-assembly storage area is configured to only store data portions of packets in said first communication flow; and

storing one or more headers from said header portion in a header storage area.

Count #19 (corresponds to Claim 19 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol, wherein said transferring comprises:

if said first packet is smaller than a predetermined threshold, storing said first packet in a first storage area; and

if said first packet is larger than said predetermined threshold, storing said first packet in a second storage area; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #20 (corresponds to Claim 20 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

determining whether a second packet received from said network is part of said first communication flow.

Count #21 (corresponds to Claim 21 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

determining whether a second packet received from said network is part of said first communication flow, wherein said determining comprises:

> maintaining a packet memory configured to store one or more packets received from said network;
>
> maintaining a flow memory configured to store, for each of said one or more packets, an identifier of a communication flow comprising said packet; and
>
> searching said flow memory for a first identifier of said first communication flow.

Count #22 (corresponds to Claim 22 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

determining whether a second packet received from said network is part of said first communication flow, wherein said determining comprises:

maintaining a packet memory configured to store one or more packets received from said network;

maintaining a flow memory configured to store, for each of said one or more packets, an identifier of a communication flow comprising said packet; and

searching said flow memory for a first identifier of said first communication flow, wherein said first identifier comprises said flow key.

Count #23 (corresponds to Claim 23 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

determining whether a second packet received from said network is part of said first communication flow, wherein said determining comprises:

maintaining a packet memory configured to store one or more packets received from said network;

maintaining a flow memory configured to store, for each of said one or more packets, an identifier of a communication flow comprising said packet; and

searching said flow memory for a first identifier of said first communication flow, wherein said first identifier comprises a flow number of said first packet, wherein said flow number is an index of said flow key within a flow database.

Count #24 (corresponds to Claim 27 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet; and

alerting said host computer system to the arrival of said first packet.

Count #25 (corresponds to Claim 28 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

maintaining a packet memory configured to store packets received from said network; and

randomly discarding a packet if said packet memory contains a pre-determined level of traffic.

Count #26 (corresponds to Claim 31 of the '360 patent):

A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol;

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet;

wherein said communication device is a network interface.

Count #27 (corresponds exactly to Claim 32 of the '360 patent):

A method of transferring a packet received at a network interface to a host computer system, comprising:

receiving a packet from a network;

storing said packet in a packet memory;

parsing a header portion of said packet;

extracting a value stored in said header portion;

identifying a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

determining whether a second packet in said packet memory is part of said communication flow;

if the host computer system contains a plurality of processors, identifying a processor to process said packet; and

storing said packet in a host memory area.

Count #28 (corresponds exactly to Claim 33 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer.

Count #29 (corresponds to Claim 34 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity, wherein said parsing comprises executing a series of updateable instructions configured to parse a packet header conforming to one of a set of pre-selected protocols;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer.

Count #30 (corresponds to Claim 35 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer;

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer; and

storing said operation code in a control memory.

Count #31 (corresponds to Claim 36 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer;

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer; and

storing a flow number of said packet in a flow memory, wherein said flow number comprises an index of said flow key in said database.

Count #32 (corresponds to Claim 37 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer;

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer;

storing a flow number of said packet in a flow memory, wherein said flow number comprises an index of said flow key in said database; and

indicating whether said packet memory includes another packet with said flow number or said flow key.

Count #33 (corresponds to Claim 38 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer;

wherein the host computer system comprises multiple processors, further comprising identifying a first processor in the host computer system to process said packet in accordance with said pre-selected protocol.

Count #34 (corresponds to Claim 39 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer;

wherein the host computer system comprises multiple processors, further comprising identifying a first processor in the host computer system to process said packet in accordance with said pre-selected protocol, further comprising:

receiving a second packet at said network interface, wherein said second packet is part of a second communication flow; and

identifying a second processor in the host computer system to process said second packet.

Count #35 (corresponds to Claim 40 of the '360 patent):

A method of transferring a packet received at a network interface from a network to a host computer system, comprising:

receiving a packet from a network at a network interface for a host computer system;

parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;

generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;

determining whether a header in said header portion conforms to a pre-selected protocol;

storing said flow key in a database;

associating an operation code with said packet, wherein said operation code identifies a status of said packet;

storing said packet in a packet memory;

if said header conforms to said pre-selected protocol:

storing a data portion of said packet in a re-assembly buffer; and

storing said header portion in a header buffer;

if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer; and

informing said host computer system of said receipt of said packet.

Count #36 (corresponds exactly to Claim 42 of the '360 patent):

An apparatus for transferring a packet to a host computer system, comprising:

a traffic classifier configured to classify a first packet received from a network by a communication flow that includes said first packet;

a packet memory configured to store said first packet;

a packet batching module configured to determine whether another packet in said packet memory belongs to said communication flow; and

a flow re-assembler configured to re-assemble a data portion of said first packet with a data portion of a second packet in said communication flow;

wherein said first packet data portion and said second packet data portion are stored in a host computer memory area to enable efficient transfer of said memory area contents.

Count #37 (corresponds to Claim 43 of the '360 patent):

An apparatus for transferring a packet to a host computer system, comprising:

a traffic classifier configured to classify a first packet received from a network by a communication flow that includes said first packet, wherein said traffic classifier comprises:

a parser configured to parse a header portion of said first packet;

a flow database configured to store a flow key identifying said communication flow; and

a flow database manager configured to manage said flow database;

wherein said flow key is generated from an identifier of a source of said first packet and an identifier of a destination of said first packet;

a packet memory configured to store said first packet;

a packet batching module configured to determine whether another packet in said packet memory belongs to said communication flow; and

a flow re-assembler configured to re-assemble a data portion of said first packet with a data portion of a second packet in said communication flow;

wherein said first packet data portion and said second packet data portion are stored in a host computer memory area to enable efficient transfer of said memory area contents.

Count #38 (corresponds to Claim 44 of the '360 patent):

A computer system for receiving a packet from a network, comprising:

a memory configured to store packets received from a network; and

a communication device configured to receive a first packet from said network, the communication device comprising:

a parser configured to extract information from a header portion of a first packet;

a flow manager configured to examine said information;

a flow database configured to store an identifier of a first communication flow comprising multiple packets, including said first packet; and

a re-assembler for storing data portions of said multiple packets in a first portion of said memory; and

a processor for processing said first packet.

Count #39 (corresponds to Claim 45 of the '360 patent):

An apparatus for transferring a packet to a host computer system, comprising:

a traffic classifier configured to classify a first packet received from a network by a communication flow that includes said first packet;

a packet memory configured to store said first packet;

a packet batching module configured to determine whether another packet in said packet memory belongs to said communication flow; and

a flow re-assembler configured to re-assemble a data portion of said first packet with a data portion of a second packet in said communication flow;

wherein said first packet data portion and said second packet data portion are stored in a host computer memory area to enable efficient transfer of said memory area contents, further comprising:

> a load distributor for identifying a first processor within the host computer system for processing said first packet and said second packet;
>
> wherein said load distributor identifies a second processor in the host computer system for processing a packet from a different communication flow.

Count #40 (corresponds exactly to Claim 47 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code.

Count #41 (corresponds to Claim 48 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code;

wherein the device is a network interface.

Count #42 (corresponds to Claim 49 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow, said flow memory comprising a flow database configured to store a flow key, wherein said flow key is assembled from an identifier of a source of said packet and an identifier of a destination of said packet;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code.

Count #43 (corresponds to Claim 50 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow, wherein said flow manager is further configured to update said flow memory as additional packets in said communication flow are received from the network;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code.

Count #44 (corresponds to Claim 51 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow said flow memory comprising a flow memory configured to store a flow number, wherein said flow number comprises an index of said communication flow in a flow database;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code.

Count #45 (corresponds to Claim 52 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code;

further comprising a control memory configured to store said operation code.

Count #46 (corresponds exactly to Claim 53 of the '360 patent):

A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet

from a communication device to a host computer, the method comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a flow key to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.

Count #47 (corresponds to Claim 54 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code;

wherein said host computer system is a multi-processor host computer system, further comprising a load distributor configured to select one of said multiple processors for processing said packet in accordance with one of said communication protocols.

Count #48 (corresponds to Claim 55 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet; and

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code;

wherein said transfer module is configured to transfer a data portion of said packet into one of a set of host memory areas in accordance with said operation code.

Count #49 (corresponds to Claim 56 of the '360 patent):

A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:

a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:

determining whether a header within said header portion conforms to one of a set of communication protocols; and

if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;

a flow memory configured to store a flow identifier for identifying said communication flow;

a flow manager configured to assign an operation code to said packet, wherein said operation code:

indicates a status of said packet; and

indicates a manner of transferring said packet to the host computer system;

a packet memory configured to store said packet;

a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance with said operation code; and

a packet batching module configured to determine whether said packet memory contains another packet in said communication flow.

Count #50 (corresponds exactly to Claim 58 of the '360 patent):

An apparatus for transferring a packet from a network to a host computer system, comprising:

a parser module configured to:

parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet;

generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and

determine whether a header in said header portion conforms to a pre-selected protocol;

a flow database configured to store said flow key;

a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;

a packet memory configured to store said first packet; and

a transfer module configured to:

if said header conforms to said pre-selected protocol:

store a data portion of said first packet in a re-assembly buffer; and

store said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol,

store said packet in a non-re-assembly buffer.

Count #51 (corresponds to Claim 59 of the '360 patent):

An apparatus for transferring a packet from a network to a host computer system, comprising:

a parser module configured to:

parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet;

generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and

determine whether a header in said header portion conforms to a pre-selected protocol;

a flow database configured to store said flow key;

a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;

a packet memory configured to store said first packet; and

a transfer module configured to:

if said header conforms to said pre-selected protocol:

store a data portion of said first packet in a re-assembly buffer; and

store said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol,

store said packet in a non-re-assembly buffer;

wherein said transfer module comprises a re-assembly engine configured to re-assemble, in said re-assembly buffer, a data portion of said first packet with a data portion of a second packet in said first communication flow.

Count #52 (corresponds to Claim 60 of the '360 patent):

An apparatus for transferring a packet from a network to a host computer system, comprising:

a parser module configured to:

parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet;

generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and

determine whether a header in said header portion conforms to a pre-selected protocol;

a flow database configured to store said flow key;

a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;

a packet memory configured to store said first packet; and

a transfer module configured to:

if said header conforms to said pre-selected protocol:

store a data portion of said first packet in a re-assembly buffer; and

store said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol,

store said packet in a non-re-assembly buffer;

further comprising a flow memory configured to store a flow number associated with said first packet, wherein said flow number comprises an index of said flow key in said flow database.

Count #53 (corresponds exactly to Claim 61 of the '360 patent):

An apparatus for transferring a packet from a network to a host computer system, comprising:

a parser module configured to:

parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet;

generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and

determine whether a header in said header portion conforms to a pre-selected protocol;

a flow database configured to store said flow key;

a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;

a packet memory configured to store said first packet; and

a transfer module configured to:

if said header conforms to said pre-selected protocol:

store a data portion of said first packet in a re-assembly buffer; and

store said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol,

store said packet in a non-re-assembly buffer;

further comprising:

a load distributor configured to identify a first processor in said host computer system for processing said first packet, said first processor being identified on the basis of said flow key;

wherein said host computer system is a multi-processor computer system; and

wherein a second processor in said host computer system is identified for processing a packet from a communication flow other than said first communication flow.

Count #54 (corresponds to Claim 62 of the '360 patent):

An apparatus for transferring a packet from a network to a host computer system, comprising:

a parser module configured to:

parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet;

generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and

determine whether a header in said header portion conforms to a pre-selected protocol;

a flow database configured to store said flow key;

a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;

a packet memory configured to store said first packet; and

a transfer module configured to:

if said header conforms to said pre-selected protocol:

store a data portion of said first packet in a re-assembly buffer; and

store said header portion in a header buffer; and

if said header conforms to a protocol other than said pre-selected protocol,

store said packet in a non-re-assembly buffer.

further comprising:

a packet batching module configured to determine whether said packet memory includes another packet in said first communication flow.

Per rule 607(a)(3), the claims in the patent (6,453,360) correspond to the counts as set forth in the table below. Per rule 607(a)(4), the claims in the present application correspond exactly to the counts as set forth in the table below:

Count No.	Corresponding Claim in USP 6,453,360	Corresponding Claim in the present application
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15
16	16	16
17	17	17
18	18	18
19	19	19
20	20	20
21	21	21
22	22	22

23	23	23
24	27	27
25	28	28
26	31	31
27	32	32
28	33	33
29	34	34
30	35	35
31	36	36
32	37	37
33	38	38
34	39	39
35	40	40
36	42	42
37	43	43
38	44	44
39	45	45
40	47	47
41	48	48
42	49	49
43	50	50
44	51	51
45	52	52
46	53	53
47	54	54
48	55	55
49	56	56
50	58	58

51	59	59
52	60	60
53	61	61
54	62	62

Per rule 607(a)(5), application of the terms of Claims 1-23, 27-28, 31-40, 42-45, 47-56 and 58-62 to the disclosure of the above-identified application is set forth in the claims chart below.

USP 6,453,360 Claim Element	Support in Alacritech Application Serial No. 60/098,296 (Page #: Line #)
1. A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:	(1:17-18) "When ***network packet arrives*** at a typical network interface card (NIC), the NIC moves the data into pre-allocated network buffers in system main memory." (3:25-26) "Alacritech was formed with the idea that the network processing described above could be offloaded onto a cost-effective Intelligent Network Interface Card (INIC)."
parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;	(6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup." (6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc)."
generating a flow key to identify a first communication flow that includes said first packet;	(6:10-14) "**CCBs** are ***initialized*** by the host during ***TCP connection setup.***" (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection.*** This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.*** (4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports.***"
transferring said first packet to a host computer system for processing in accordance with	(5:25-27) "In the slow-path case, ***network frames are handed to the system*** at the MAC layer and passed up through the host protocol stack like any other network frame."

said pre-selected protocol; and	(83:8-9) “When the INIC receives a frame, one of its immediate tasks is to determine if the frame is for a CCB that it controls. If not, ***the frame is passed to the host*** on what is termed the slow path.”
associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.	(6:36-37) “The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***.” (71:14-15) “These ***status words*** are generated by the INIC hardware and ***placed in front of the receive frame***.”
2. The method of claim 1, wherein said parsing comprises:	
copying a header portion of said first packet into a header memory; and	(13:17-18) “As mentioned above, the fast-path flow puts a ***header*** into a ***header buffer*** that is then forwarded to the host.”
examining said header portion according to a series of parsing instructions;	(160:2-3) “The receive sequencer (RcvSeq) ***analyzes*** and manages incoming packets, stores the result in dram buffers, then notifies the processor…” (85:26-38) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue. The pointer contains a bit (bit 29) that informs the ***microcode*** if this frame is definitely not a fast-path candidate (e.g., not TCPIP, or has an error of some sort). Receive frame processing involves extracting this pointer from the Receive hardware queue, and setting up a DMA into an SRAM header buffer of the first X bytes from the DRAM frame buffer. The size of the DMA is determined by whether bit 29 is set or not. If it is set (this frame is not a fast-path candidate), then only the status bytes are needed by the microcode, so the size would be 16 bytes. Otherwise up to 92 bytes are DMA’d – sufficient to get all useful headers. When this DMA is complete, the status bytes are used by the ***microcode*** to determine whether to jump to fast-path or slow-path processing. If bit 29 is set, this frame is going slow-path. Effectively this means that the frame will not be processed against an on-INIC CCB. It will be passed directly to the host, although if the frame is TCPIP, then its checksums have already been validated by the hardware. Also, all other header validations have been performed.”
wherein said parsing instructions are configured to reflect a set of pre-selected communication protocols.	(85:15-28) “Much ***header validation*** is implemented in hardware in conjunction with MAC processing by scanning the data as it flies by. The following tests are performed and appropriate status is generated by the ***Receive hardware sequencer***. • MAC header: determine if Ethernet/802.3, if MC/BC, if it matches our MAC address A or B, determine the network protocol, flag if not a MAC status of “good packet.”

	• Network header: determine if header checksum is valid, header length is valid (e.g. IP >= 5), network length > header length, what the transport protocol is, if there is any fragmentation or network options, destination network address (one of ours?) • Transport header: checksum is valid (incl pseudo-header if relevant), header length is valid (e.g. TCP >= 5), length is valid, what is the session layer protocol (e.g. SMB, HTTP or FTP data), are there any transport flags set (e.g. FIN/SYN/URG/RST bits), and any options present."
3. The method of claim 2, wherein said parsing instructions are updateable.	(104:11) "Writable control store (WCS) allows field ***updates*** for feature enhancements." (62:28) "The first step in initializing the card is to reset and download the card." (63:1) "The download is performed by a series of three writes to the WCS register on the INIC." (63:8-9) "We first download all of the code, and then we download it all again with the bit 30 set in the address words."
4. The method of claim 2, further comprising copying a value from a field in a header of said header portion.	(72:30-32) "The other way is to set the first 6 bytes of the header buffer (the destination MAC address) to our interfaces MAC address. While this does require a 6-byte ***copy*** for every frame received, this was determined to be the best approach."
5. The method of claim 1, wherein said parsing comprises: extracting an identifier of a source of said first packet from said header portion; and extracting an identifier of a destination of said first packet from said header portion.	(6:36-38) "The header is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the ***IP addresses and TCP ports*** to expedite CCB lookup." (160:6-7) "The ***Mac, network, transport and session information is analyzed*** as each byte is received and stored in the assembly register (AssyReg)." (86:1-3) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***.
6. The method of claim 5, wherein said generating comprises combining said source identifier and said destination identifier.	(6:10-14) "**CCBs** are ***initialized*** by the host during ***TCP connection setup***." (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection***. (4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports***." (86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and***

	destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***"
7. The method of claim 1, wherein said generating comprises retrieving an identifier of a communication connection from said header portion.	See support for claim 1 above.
8. The method of claim 1, further comprising storing said first packet in a packet memory prior to said transferring.	(84:6-7) "1. Receive input queue: hardware will automatically ***DMA arriving frames into frame buffers*** and queue an event into the Q_RECV hardware queue."
9. The method of claim 1, further comprising storing said flow key in a flow database, wherein said flow database is configured to facilitate management of said first communication flow.	(6:10-16) "This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The ***complete set of CCBs exists in host memory***, but ***a subset of these may be "owned" by the card*** at any given time. This subset is the ***CCB cache***. The INIC can own up to 256 CCBs at any given time."
10. The method of claim 9, further comprising associating a flow number with said first packet, wherein said flow number comprises an index of said flow key within said flow database.	(86:1-8) "The receive sequencer has already generated a ***hash*** based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This ***hash*** is used to ***index*** directly into a hash table on the INIC that ***points to entries in a CCB header table***. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame." (24:36-39) "The initial command from ATCP to INIC expresses an "intention" to hand out the context. It carries a ***context number***; context numbers are allocated by the ATCP driver, which keeps a per-INIC ***table of free and in-use context numbers***. It also includes the ***source and destination IP addresses and ports***, which will allow the INIC to establish a "provisional" context."
11. The method of claim 10, further comprising storing said flow number in a flow memory.	Part of FRAME Status B is the hash. (164:15-17) "07:00 CtxHsh The 8-bit context-***hash*** generated by exclusive-oring all bytes of the ***IP source address, IP destination-address, transport source port and the transport destination port***."

	(80:13-16) "A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, moves the frame into a DRAM buffer and ***queues*** the frame address and ***status*** for processing by the Receive CPU into one of the hardware queues mentioned above." (86:3-4) "This ***hash*** is used to index directly into a ***hash table*** on the INIC that points to entries in a CCB header table." (24:37-38) "***context numbers*** are allocated by the ATCP driver, which keeps a per-INIC table of free and in-use context numbers."
12. The method of claim 9, further comprising updating an entry in said flow database associated with said flow key when a second packet in said first communication flow is received.	(86:7-8) "If a ***match is found***, then the frame will be ***processed against the CCB*** by the INIC." (6:10-16) "A ***CCB*** is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the ***current send and receive sequence numbers***..."
13. A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet received at a network interface from a network to a host computer system, the method comprising:	(110:12) "The ***processor instructions reside in the on-chip control-store, which is implemented as a mixture of ROM and Sram.***" (80:6-31) "As specified in other sections, the INIC supplies a set of 3 custom processors (CPUs) that provide considerable hardware-assist to the microcode running thereon... • Multiple register contexts or process slots with register access controlled by simply setting a process register . The Protocol Processor will provide 512 SRAM-based registers to be shared among the 3 CPUs in any way desired. The current implementation uses 16 processes of 16 registers each, leaving 256 scratch registers to be shared. • A set of CPU-specific registers that are the same local-cpu register number, but for which the real register is determined by an offset based on the CPU number; this allows multiple CPUs to execute the same code at the same time without register clashes or interlocks. These registers are a part of the above-mentioned scratch pool."
receiving a packet from a network at a network interface for a host computer system;	(80:12-13) "***Receive Frame Processing*** Receive-frame processing can be broken down into the following stages"
parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;	(6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup." (160:6-7) "The ***Mac, network, transport and session information is analyzed*** as each byte is received and stored in the assembly register (AssyReg)." (86:1-3) "The receive sequencer has already ***generated*** a hash based

	on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***.
generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;	(6:10-14) “**CCBs** are ***initialized*** by the host during ***TCP connection setup***.” (6:10-12) “A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection***. (4:16-19) “The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports***.”
determining whether a header in said header portion conforms to a pre-selected protocol;	(6:30-32) “When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc).”
storing said flow key in a database;	(6:14-16) “The complete set of ***CCBs exists in host memory***, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time.” (6:10-12) “A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection***.”
associating an operation code with said packet, wherein said operation code identifies a status of said packet;	(6:36-38) “The header is fully parsed by hardware and its type is summarized in a single ***status word***.
storing said packet in a packet memory;	(80:13-17) “A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, ***moves the frame into a DRAM buffer*** and queues the frame address and status for processing by the Receive CPU into one of the hardware queues mentioned above.”
if said header conforms to said pre-selected protocol:	(41:25) “2. If the type field contains our custom INIC type (TCP for example):”
storing a data portion of said packet in a re-assembly buffer; and	(5:27-28) “In the fast path case, network data is given to the host after the headers have been processed and stripped.” See diagram on page 13. (14:27) “Receive data buffers are allocated in blocks of 2, 2k bytes each (4k page).”

	(22:26-28) "The INIC moves succeeding incoming segments into the MDL: and since the granting of the MDL may be regarded as a "promise" by the TDI client to accept the data, the INIC does not have to close its advertised receive window while filling the MDL."
storing said header portion in a header buffer; and	(13:17-18) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host."
if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer.	(11:2-10) "If the INIC receives a frame that does not contain a TCP segment for one of its CCB's, it simply passes it to the host as if it were a dumb NIC. If the frame fits into a small buffer (~200 bytes or less), then it simply fills in the small buffer with the data and notifies the host. Otherwise it places the data in a large buffer, writes the address of the large buffer into a small buffer, and again notifies the host. The host, having received the interrupt and found the completed small buffer, checks to see if the data is contained in the small buffer, and if not, locates the large buffer. Having found the data, the host will then pass the frame upstream to be processed by the standard protocol stack. It must also replenish the INIC's small and large receive buffer pool if necessary."
14. The method of claim 1, wherein said associating comprises:	
retrieving one or more header fields of said header portion; and analyzing said header fields to determine said status of said first packet.	(85;15-28) "Much ***header validation*** is implemented in hardware in conjunction with MAC processing by scanning the data as it flies by. The following tests are performed and appropriate ***status is generated*** by the Receive hardware sequencer: • MAC header: determine if Ethernet/802.3, if MC/BC, if it matches our MAC address A or B, determine the network protocol, flag if not a MAC status of "good packet." • Network header: determine if header checksum is valid, header length is valid (e.g. IP >= 5), network length > header length, what the transport protocol is, if there is any fragmentation or network options, destination network address (one of ours?) • Transport header: checksum is valid (incl pseudo-header if relevant), header length is valid (e.g. TCP >= 5), length is valid, what is the session layer protocol (e.g. SMB, HTTP or FTP data), are there any transport flags set (e.g. FIN/SYN/URG/RST bits), and any options present. As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above

	frame status.” (6:36-37) “The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***.” (71:14-15) “These ***status words*** are generated by the INIC hardware and ***placed in front of the receive frame***.”
15. The method of claim 14, wherein said analyzing comprises:	
determining whether said first packet includes a data portion; and	(52:24-25) “when we detect an incoming SYN segment in tcp_input(), we call ATKPassiveConnect(), which is located in the file atktdi.c.”
if said first packet includes a data portion, determining whether said data portion exceeds a pre-determined size.	(163:45) “29 BufOflw The frame length exceeded the capacity of the current buffer.” (7:28-36) “Clearly there are circumstances in which this does not make sense. When a small amount of data (500 bytes for example), with a push flag set indicating that the data must be delivered to the client immediately, it does not make sense to deliver some of the data directly while waiting for the list of addresses to DMA the rest. Under these circumstances, it makes more sense to deliver the 500 bytes directly to the host, and allow the host to copy it into its final destination. While various ranges are feasible, it is currently preferred that anything less than a segment’s (1500 bytes) worth of data will be delivered directly to the host, while anything more will be delivered as a small piece (which may be 128 bytes), while waiting until receiving the destination memory address before moving the rest.”
16. The method of claim 14, wherein said analyzing comprises determining whether said first packet was received out of order in said first communication flow.	(6:19-24) “From this point on, the INIC owns the connection until either a FIN arrives signaling that the connection is being closed, or until an exception occurs which the INIC is not designed to handle (such as an ***out of order segment***). When any of these conditions occur, the INIC will then flush the CCB back to host memory, and issue a message to the host telling it that it has relinquished control of the connection, thus putting the connection back into slow-path mode.” (163:46) “28 PktMissed A ***frame was missed prior to receiving the current frame***.”
17. The method of claim 1, further comprising storing said operation code in a control memory.	(85:27-29) “As frames are received, they are ***placed into*** 2K-byte DRAM ***buffers*** by the Receive hardware sequencer, ***along with*** 16 bytes of the above frame ***status***. A pointer to the last byte + 1 of this buffer is ***queued into*** the Q_RECV ***queue***.”
18. The method of claim 1, wherein said first packet is determined to conform to said pre-selected protocol, said transferring comprising:	(41:25-27) “2. If the type field contains our custom INIC type (TCP for example): A. If the header buffer specifies a fast-path connection, allocate one or more mbufs headers to map the header and possibly data buffers.”

storing a data portion of said first packet in a re-assembly storage area, wherein said re-assembly storage area is configured to only store data portions of packets in said first communication flow; and	(5:27-28) "In the fast path case, network data is given to the host after the headers have been processed and stripped." (7:23-27) "We will make use of this feature by providing a small amount of any received data to the host, with a notification that we have more data pending. When this small amount of data is passed up to the client, and it returns with the address in which to put the remainder of the data, our host transport driver will pass that address to the INIC which will DMA the remainder of the data into its final destination."
storing one or more headers from said header portion in a header storage area.	(13:17-18) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host."
19. The method of claim 1, wherein said transferring comprises:	
if said first packet is smaller than a predetermined threshold, storing said first packet in a first storage area; and if said first packet is larger than said predetermined threshold, storing said first packet in a second storage area.	(14:6-7) "***Header buffers*** in host memory are 256 bytes long, and are aligned on 256 byte boundaries." (14:27) "***Receive data buffers*** are allocated in blocks of 2, 2k bytes each (4k page)." (13:4-5) "We take advantage of this fact by allocating ***large and small receive buffers. If a received frame fits in a small buffer, the INIC will use a small buffer. Otherwise it will use a large buffer.***" (11:3-8) "If the INIC receives a frame that does not contain a TCP segment for one of its CCB's, it simply passes it to the host as if it were a dumb NIC. ***If the frame fits into a small buffer (~200 bytes or less***), ***then it simply fills in the small buffer with the data*** and notifies the host. ***Otherwise it places the data in a large buffer***, writes the address of the large buffer into a small buffer, and again notifies the host. The host, having received the interrupt and found the completed small buffer, checks to see if the data is contained in the small buffer, and if not, locates the large buffer. Having found the data, the host will then ***pass the frame upstream to be processed by the standard protocol stack***. It must also replenish the INIC's small and large receive buffer pool if necessary."
20. The method of claim 1, further comprising determining whether a second packet received from said network is part of said first communication flow.	(86:1-9) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC. If not, then the

	frame is sent for slow-path processing."
21. The method of claim 20, wherein said determining comprises:	
maintaining a packet memory configured to store one or more packets received from said network;	(85:27-28) "As frames are received, they are ***placed into*** 2K-byte DRAM ***buffers*** by the Receive hardware sequencer."
maintaining a flow memory configured to store, for each of said one or more packets, an identifier of a communication flow comprising said packet; and	(6:10-16) "This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time."
searching said flow memory for a first identifier of said first communication flow.	86:1-9) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC. If not, then the frame is sent for slow-path processing."
22. The method of claim 21, wherein said first identifier comprises said flow key.	(86:6-8) "for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
23. The method of claim 21, wherein said first identifier comprises a flow number of said first packet, wherein said flow number is an index of said flow key within a flow database.	(86:2-4) "a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table."
27. The method of claim 1, further comprising alerting said host computer system to the arrival of said first packet.	(2:29-30) "Each of these segments may result in an interrupt to the CPU."

28. The method of claim 1, further comprising:	
maintaining a packet memory configured to store packets received from said network; and	(105:37-40) "External SDram provides frame buffering, which is configurable as 4MB, 8MB, 16MB or 32MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224MB/s. The buffer provides temporary storage of both incoming and outgoing frames."
randomly discarding a packet if said packet memory contains a pre-determined level of traffic.	(17:7) "RMISS Rcv drop occurred due to no buffers"
31. The method of claim 1, wherein said communication device is a network interface.	(3:25-26) "the network processing described above could be offloaded onto a cost-effective Intelligent ***Network Interface*** Card (INIC)."
32. A method of transferring a packet received at a network interface to a host computer system, comprising:	
receiving a packet from a network;	(6:30-31) "When ***a frame is received by the INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not."
storing said packet in a packet memory;	(80:13-17) "A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, ***moves the frame into a DRAM buffer*** and queues the frame address and status for processing by the Receive CPU into one of the hardware queues mentioned above." (105:37-40) "External SDram provides frame buffering, which is configurable as 4MB, 8MB, 16MB or 32MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224MB/s. The ***buffer provides temporary storage of both incoming and outgoing frames***."
parsing a header portion of said packet;	(6:36-38) "The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup."
extracting a value stored in said header portion;	(85;15-28) "Much ***header validation*** is implemented in hardware in conjunction with MAC processing by scanning the data as it flies by. The following tests are performed and appropriate status is generated by the ***Receive hardware sequencer***: • MAC header: determine if Ethernet/802.3, if MC/BC, if it

	matches our MAC address A or B, determine the network protocol, flag if not a MAC status of "good packet." • Network header: determine if header checksum is valid, header length is valid (e.g. IP >= 5), network length > header length, what the transport protocol is, if there is any fragmentation or network options, destination network address (one of ours?) • Transport header: checksum is valid (incl pseudo-header if relevant), header length is valid (e.g. TCP >= 5), length is valid, what is the session layer protocol (e.g. SMB, HTTP or FTP data), are there any transport flags set (e.g. FIN/SYN/URG/RST bits), and any options present. As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status."
identifying a communication flow comprising said packet;	(83:8-9) "When the INIC receives a frame, one of its immediate tasks is to ***determine if the frame is for a CCB that it controls***."
determining whether a header in said header portion conforms to a pre-selected protocol;	(6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc)."
determining whether a second packet in said packet memory is part of said communication flow;	(86:1-8) "The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to ***determine if a CCB exists on the INIC for this frame***. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
if the host computer system contains a plurality of processors, identifying a processor to process said packet; and	(4:2-3) "In order to keep the system ***CPU*** from having to process the packet headers or checksum the packet, we must perform this task on the ***INIC***." (105:1-4) "7.1. ***INIC*** General Description The ***microprocessor*** is a 32-bit, full-duplex, four channel, 10/100-Megabit per second (Mbps), Intelligent Network Interface Controller, designed to provide high-speed protocol processing for server applications...The ***microprocessor*** can be used by ***PCs, workstations*** and ***routers*** or anywhere that TCP/IP protocols are being utilized." (80:6) "the ***INIC supplies a set of 3 custom processors (CPUs)***..." (80:28-31) "A set of CPU-specific registers that are the same local-cpu register number, but for which the real register is determined by an offset

	based on the CPU number; ***this allows multiple CPUs to execute the same code at the same time*** without register clashes or interlocks.” (85:26-30) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate...” (85:38) “***If bit 29 is set, this frame is going slow-path.***”
storing said packet in a host memory area.	(5:25-27) “In the slow-path case, network frames are handed to the system at the MAC layer and passed up through the host protocol stack like any other network frame.”
33. A method of transferring a packet received at a network interface from a network to a host computer system, comprising:	
receiving a packet from a network at a network interface for a host computer system;	(6:30-31) “When ***a frame is received by the INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not.”
parsing a header portion of said packet to extract an identifier of a source entity and an identifier of a destination entity;	(6:36-38) “The ***header*** is fully ***parsed*** by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup.”
generating a flow key from said source identifier and said destination identifier to identify a communication flow comprising said packet;	(6:10-14) “**CCBs** are ***initialized*** by the host during ***TCP connection setup.***” (6:10-12) “A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection***. (4:16-19) “The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports.***”
determining whether a header in said header portion conforms to a pre-selected protocol;	(6:30-32) “When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc).”
storing said flow key in a database;	(6:10-16) “This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the

	connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The ***complete set of CCBs exists in host memory***, but a subset of these may be "owned" by the card at any given time. This subset is the ***CCB cache***. The INIC can own up to 256 CCBs at any given time."
associating an operation code with said packet, wherein said operation code identifies a status of said packet;	(6:36-37) "The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***." (71:14-15) "These ***status words*** are generated by the INIC hardware and ***placed in front of the receive frame***."
storing said packet in a packet memory;	(80:13-17) "A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, ***moves the frame into a DRAM buffer*** and queues the frame address and status for processing by the Receive CPU into one of the hardware queues mentioned above." (105:37-40) "External SDram provides frame buffering, which is configurable as 4MB, 8MB, 16MB or 32MB using the appropriate SIMMs. Use of -10 speed grades yields an external buffer bandwidth of 224MB/s. The buffer provides temporary storage of both incoming and outgoing frames."
if said header conforms to said pre-selected protocol: storing a data portion of said packet in a re-assembly buffer; and storing said header portion in a header buffer; and	(37:31-38:2) "1. Packets delivered to ATCP only (not accepted by MSTCP): TCP, TTCP or SPX packets destined for one of our IP addresses. This includes both slow-path frames and fast-path frames. In the slow-path case, the TCP frames are given in their entirety (headers included). In the fast-path case, the ATKReceivePacket is given a header buffer that contains status information and data with no headers (except those above TCP)."
if said header conforms to a protocol other than said pre-selected protocol, storing said packet in a non-re-assembly buffer.	(41:34-36) "3. If the type field of the frame indicates ARP or ICMP: A. Allocate a mbuf with a data buffer. Copy the contents of the packet into the mbuf. Queue the mbuf, and return 0 (not accepted)."
34. The method of claim 33, wherein said parsing comprises executing a series of updateable instructions configured to parse a packet header conforming to one of a set of pre-selected protocols	(62:28) "The first step in initializing the card is to reset and download the card." (63:1) "The download is performed by a series of three writes to the WCS register on the INIC." (63:8-9) "We first download all of the code, and then we download it all again with the bit 30 set in the address words." (85:16-26) "The following tests are performed and appropriate status is generated by the Receive hardware sequencer:

	• MAC header: determine if Ethernet/802.3, if MC/BC, if it matches our MAC address A or B, determine the network protocol, flag if not a MAC status of "good packet." • Network header: determine if header checksum is valid, header length is valid (e.g. IP >= 5), network length > header length, what the transport protocol is, if there is any fragmentation or network options, destination network address (one of ours?) • Transport header: checksum is valid (incl pseudo-header if relevant), header length is valid (e.g. TCP >= 5), length is valid, what is the session layer protocol (e.g. SMB, HTTP or FTP data), are there any transport flags set (e.g. FIN/SYN/URG/RST bits), and any options present."
35. The method of claim 33, further comprising storing said operation code in a control memory.	(6:36-37) "The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***." (71:14-15) "These ***status words*** are generated by the INIC hardware and placed in front of the receive frame." (80:13-17) "A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, moves the frame into a DRAM buffer and ***queues the frame address and status*** for processing by the Receive CPU into ***one of the hardware queues*** mentioned above."
36. The method of claim 33, further comprising storing a flow number of said packet in a flow memory, wherein said flow number comprises an index of said flow key in said database.	(6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection***. (4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports***." (86:1-8) "The receive sequencer has already generated a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to ***index*** directly into a hash table on the INIC that points to ***entries in a CCB header table***. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a ***CCB exists on the INIC for this frame***. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
37. The method of claim 36, further comprising indicating whether said packet memory includes another packet with said flow number or said flow key.	(21:25-27) "Once the INIC has passed to the host an indication with an NETBIOS length greater than the amount of data in the packet it passed, it goes into a state where it is expecting an MDL from the host; in this state, any further incoming data is accumulated in DRAM on the INIC."
38. The method of claim 33, wherein the host computer	(80:6-31) "As specified in other sections, the INIC supplies a set of 3 custom processors (CPUs) that provide considerable hardware-assist to

system comprises multiple processors, further comprising identifying a first processor in the host computer system to process said packet in accordance with said pre-selected protocol.	the microcode running thereon." (85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:38) "***If bit 29 is set, this frame is going slow-path.***"
39. The method of claim 38, further comprising:	
receiving a second packet at said network interface, wherein said second packet is part of a second communication flow; and	
identifying a second processor in the host computer system to process said second packet.	See support for claim 38 above. For claim 39, however, second processor may be an INIC or CPD processor. For a description of the fast-path, see for example (B:10,16-end).
40. The method of claim 33, further comprising informing said host computer system of said receipt of said packet.	(2:29-30) "Each of these segments may result in an interrupt to the CPU."
42. An apparatus for transferring a packet to a host computer system, comprising:	
a traffic classifier configured to classify a first packet received from a network by a communication flow that includes said first packet;	(80:13-17) "A Receive hardware sequencer that completely validates an input header as the frame is being received by the MAC, validates TCP and IP checksums, generates a frame status and a context lookup hash, moves the frame into a DRAM buffer and queues the frame address and status for processing by the Receive CPU into one of the hardware queues mentioned above."
a packet memory configured to store said first packet;	(85:27-28) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status."
a packet batching module configured to determine whether another packet in said packet memory belongs to said communication flow; and	(86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its***

	source and destination addresses and ports with those of the frame."
a flow re-assembler configured to re-assemble a data portion of said first packet with a data portion of a second packet in said communication flow;	(14:1-3) "Note that the order in which data is written is important. Data buffers are moved by DMA into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived."
wherein said first packet data portion and said second packet data portion are stored in a host computer memory area to enable efficient transfer of said memory area contents.	(13:17-21) "As mentioned above, the fast-path flow puts a header into a header buffer that is then forwarded to the host. The host uses the header to determine what further data is following, allocates the necessary host buffers, and these are passed back to the INIC via a command to the INIC. The INIC then fills these buffers from data it was accumulating on the card and notifies the host by sending a response to the command."
43. The apparatus of claim 42, wherein said traffic classifier comprises:	
a parser configured to parse a header portion of said first packet;	(6:36-37) "The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***."
a flow database configured to store a flow key identifying said communication flow; and	(6:10-16) "This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time."
a flow database manager configured to manage said flow database;	(79:31) "LRU Least Recently Used; used in the SRAM CCB buffer cache."
wherein said flow key is generated from an identifier of a source of said first packet and an identifier of a destination of said first packet.	(6:10-14) "**CCBs** are ***initialized*** by the host during ***TCP connection setup.***" (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.*** (4:16-19) "The ... ***context*** is ... ***identified by the IP source and***

	destination addresses and TCP source and destination ports.” (86:1-3) “The receive sequencer has already generated a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports.***
44. A computer system for receiving a packet from a network, comprising:	
a memory configured to store packets received from a network; and	(85:27-28) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status.”
a communication device configured to receive a first packet from said network, the communication device comprising:	(3:25-26) “the network processing described above could be offloaded onto a cost-effective Intelligent Network Interface Card (INIC).”
a parser configured to extract information from a header portion of a first packet;	(6:36-37) “The header is fully ***parsed by hardware*** and its type is summarized in a single status word.”
a flow manager configured to examine said information;	(85:27-41) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue. The pointer contains a bit (bit 29) that informs the microcode if this frame is definitely not a fast-path candidate (e.g., not TCPIP, or has an error of some sort). Receive frame processing involves extracting this pointer from the Receive hardware queue, and setting up a DMA into an SRAM header buffer of the first X bytes from the DRAM frame buffer. The size of the DMA is determined by whether bit 29 is set or not. If it is set (this frame is not a fast-path candidate), then only the status bytes are needed by the microcode, so the size would be 16 bytes. Otherwise up to 92 bytes are DMA'd – sufficient to get all useful headers. When this DMA is complete, the status bytes are used by the microcode to determine whether to jump to fast-path or slow-path processing.”
a flow database configured to store an identifier of a first communication flow comprising multiple packets, including said first packet; and	(6:10-16) “This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time.

	This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time."
a re-assembler for storing data portions of said multiple packets in a first portion of said memory; and	(14:1-3) "Note that the order in which data is written is important. Data buffers are moved by DMA into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived."
a processor for processing said first packet.	(105:2-4) "The microprocessor is a 32-bit, full-duplex, four channel, 10/100-Megabit per second (Mbps), Intelligent Network Interface Controller, designed to provide high-speed protocol processing for server applications."
45. The apparatus of claim 42, further comprising:	
a load distributor for identifying a first processor within the host computer system for processing said first packet and said second packet;	(85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..." (85:26-30) "***If bit 29 is set, this frame is going slow-path.***"
wherein said load distributor identifies a second processor in the host computer system for processing a packet from a different communication flow.	(86:1-8) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC."
47. A device for receiving a packet from a network and transferring the packet to a host computer system, comprising:	
a parser configured to parse a header portion of a packet received from a network, wherein said parsing comprises:	(6:36-37) "The header is fully ***parsed by hardware*** and its type is summarized in a single status word."
determining whether a header within said header portion	(6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its

conforms to one of a set of communication protocols; and	CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc).”
if said header conforms to one of said communication protocols, extracting information from said header portion to identify a communication flow to which said packet belongs;	(6:36-38) “The header is fully parsed by hardware and its type is summarized in a single status word. The checksum is also verified automatically in hardware, and a hash key is created out of the IP addresses and TCP ports to expedite CCB lookup.” (86:1-8) “The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***”
a flow memory configured to store a flow identifier for identifying said communication flow;	(6:10-16) “This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time.”
a flow manager configured to assign an operation code to said packet, wherein said operation code: indicates a status of said packet; and indicates a manner of transferring said packet to the host computer system;	(85:26-30) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate...” (85:38) “***If bit 29 is set, this frame is going slow-path.***”
a packet memory configured to store said packet; and	(14:27) “Receive data buffers are allocated in blocks of 2, 2k bytes each (4k page).”
a transfer module configured to transfer said packet from said packet memory to a host computer system in accordance	(14:1-3) “Data buffers are moved by DMA into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived.”

with said operation code.	
48. The device of claim 47, wherein the device is a network interface.	(3:25-26) "the network processing described above could be offloaded onto a cost-effective Intelligent ***Network Interface*** Card (INIC)."
49. The device of claim 47, said flow memory comprising a flow database configured to store a flow key, wherein said flow key is assembled from an identifier of a source of said packet and an identifier of a destination of said packet.	(86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***"
50. The device of claim 47, wherein said flow manager is further configured to update said flow memory as additional packets in said communication flow are received from the network.	(86:8) "If a match is found, then the frame will be processed against the CCB by the INIC."
51. The device of claim 47, said flow memory comprising a flow memory configured to store a flow number, wherein said flow number comprises an index of said communication flow in a flow database.	(6:10-16) "This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time."
52. The device of claim 47, further comprising a control memory configured to store said operation code.	(85:27-29) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status. A pointer to the last byte + 1 of this buffer is queued into the Q_RECV queue."
53. A computer readable storage medium storing instructions that, when executed by a computer, cause the computer to perform a method of transferring a packet from a communication device to a host computer, the method comprising: parsing a header portion of a	(6:36-37) "The header is fully ***parsed*** by hardware and its type is

first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;	summarized in a single status word." (6:30-32) "When a ***frame is received*** by the ***INIC***, it must verify it completely before it even determines whether it belongs to one of its CCBs or not. This includes all header validation (***is it IP, IPV4 or V6***, is the IP header checksum correct, is the TCP checksum correct, etc)."
generating a flow key to identify a first communication flow that includes said first packet;	(6:10-14) "**CCBs** are ***initialized*** by the host during ***TCP connection setup***." (6:10-12) "A ***CCB*** is a structure that contains the entire ***context*** associated with a ***connection***. This includes the ***source and destination IP addresses and source and destination TCP ports*** that ***define the connection.*** (4:16-19) "The ... ***context*** is ... ***identified by the IP source and destination addresses and TCP source and destination ports***." (86:1-3) "The receive sequencer has already generated a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***.
transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and	(85:38) "***If bit 29 is set, this frame is going slow-path***."
associating an operation code with said first packet, wherein said operation code indicates a status of said first packet.	(85:26-30) "As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate..."
54. The device of claim 47, wherein said host computer system is a multi-processor host computer system, further comprising a load distributor configured to select one of said multiple processors for processing said packet in accordance with one of said communication protocols.	(86:1-8) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC."
55. The device of claim 47, wherein said transfer module is configured to transfer a data portion of said packet into one of a set of host memory areas in accordance with said operation	(14:1-3) "Note that the order in which data is written is important. Data buffers are moved by DMA into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived."

code.	
56. The device of claim 47, further comprising a packet batching module configured to determine whether said packet memory contains another packet in said communication flow.	(86:1-8) "The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
58. An apparatus for transferring a packet from a network to a host computer system, comprising:	
a parser module configured to: parse a header portion of a first packet received from a network to extract an identifier of a source of said first packet and an identifier of a destination of said first packet; generate a flow key from said source identifier and said destination identifier to identify a communication flow comprising said first packet; and determine whether a header in said header portion conforms to a pre-selected protocol;	(6:36-38) "The header is fully parsed by hardware and its ***type is summarized*** in a single ***status word***." The checksum is also verified automatically in hardware, and a hash key is created out of the ***IP addresses and TCP ports*** to expedite CCB lookup." (85:15-28) "Much ***header validation*** is implemented in hardware in conjunction with MAC processing by scanning the data as it flies by. The following tests are performed and appropriate status is generated by the ***Receive hardware sequencer***: • MAC header: determine if Ethernet/802.3, if MC/BC, if it matches our MAC address A or B, determine the network protocol, flag if not a MAC status of "good packet." • Network header: determine if header checksum is valid, header length is valid (e.g. IP >= 5), network length > header length, what the transport protocol is, if there is any fragmentation or network options, destination network address (one of ours?) • Transport header: checksum is valid (incl pseudo-header if relevant), header length is valid (e.g. TCP >= 5), length is valid, what is the session layer protocol (e.g. SMB, HTTP or FTP data), are there any transport flags set (e.g. FIN/SYN/URG/RST bits), and any options present. As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above frame status." (86:1-8) "The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash

	table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***”
a flow database configured to store said flow key;	(6:10-16) “This introduces the notion of a Communication Control Block (CCB) cache. A CCB is a structure that contains the entire context associated with a connection. This includes the source and destination IP addresses and source and destination TCP ports that define the connection. It also contains information about the connection itself such as the current send and receive sequence numbers, and the first-hop MAC address, etc. The complete set of CCBs exists in host memory, but a subset of these may be "owned" by the card at any given time. This subset is the CCB cache. The INIC can own up to 256 CCBs at any given time.”
a flow database manager configured to associate an operation code with said first packet, wherein said operation code identifies a status of said first packet;	(85:26-30) “As frames are received, they are placed into 2K-byte DRAM buffers by the Receive hardware sequencer, along with 16 bytes of the above ***frame status***. A ***pointer*** to the last byte + 1 of this buffer is queued into the Q_RECV queue. The ***pointer*** contains ***a bit (bit 29)*** that informs the microcode if this frame is definitely not a fast-path candidate...” (85:38) “***If bit 29 is set, this frame is going slow-path.***”
a packet memory configured to store said first packet; and	(14:27) “Receive data buffers are allocated in blocks of 2, 2k bytes each (4k page).”
a transfer module configured to:	(14:1-3) “Data buffers are moved by DMA into the host before the header buffer, since the header buffer contains the status word designating that the data has arrived.”
if said header conforms to said pre-selected protocol: store a data portion of said first packet in a re-assembly buffer; and store said header portion in a header buffer; and if said header conforms to a protocol other than said pre-selected protocol, store said packet in a non-re-assembly buffer.	(86:1-8) “If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC.” (21:4-6) “In slow-path, however, the INIC is operating as a conventional “dumb NIC”, and the packets passing between ATCP driver and INIC contain all the header info from MAC layer on up.”
59. The apparatus of claim 58, wherein said transfer module	(43:10-14) “First the INIC fills in a host receive buffer with a relatively small amount of data, but notifies the host of a large amount of pending

comprises a re-assembly engine configured to re-assemble, in said re-assembly buffer, a data portion of said first packet with a data portion of a second packet in said first communication flow.	data (either through a large amount of buffered data on the card, or through a large amount of expected NetBios data). This small amount of data is delivered to the client through the TDI interface. The client then responds with a MDL in which the data should be placed."
60. The apparatus of claim 58, further comprising a flow memory configured to store a flow number associated with said first packet, wherein said flow number comprises an index of said flow key in said flow database.	(86:1-8) "The receive sequencer has already generated a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to ***index*** directly into a hash table on the INIC that points to ***entries in a CCB header table***. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a ***CCB exists on the INIC for this frame***. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame."
61. The apparatus of claim 58, further comprising:	
a load distributor configured to identify a first processor in said host computer system for processing said first packet, said first processor being identified on the basis of said flow key;	(86:1-8) "If bit 29 is not set, then there may be an onboard CCB for this frame. The receive sequencer has already generated a hash based on the network and transport addresses, e.g., IP source and destination addresses and TCP ports. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, comparing its source and destination addresses and ports with those of the frame. If a match is found, then the frame will be processed against the CCB by the INIC."
wherein said host computer system is a multi-processor computer system; and	(4:2-3) "In order to keep the system ***CPU*** from having to process the packet headers or checksum the packet, we must perform this task on the ***INIC***." (105:1-4) "7.1. ***INIC*** General Description The ***microprocessor*** is a 32-bit, full-duplex, four channel, 10/100-Megabit per second (Mbps), Intelligent Network Interface Controller, designed to provide high-speed protocol processing for server applications...The ***microprocessor*** can be used by ***PCs, workstations*** and ***routers*** or anywhere that TCP/IP protocols are being utilized." (80:6) "the ***INIC supplies a set of 3 custom processors (CPUs)***..." (80:28-31) "A set of CPU-specific registers that are the same local-cpu register number, but for which the real register is determined by an offset based on the CPU number; ***this allows multiple CPUs to execute the same code at the same time*** without register clashes or interlocks."
wherein a second processor in said host computer system is identified for processing a packet from a communication flow other than said first	(85:38-86:1) "If bit 29 is set, this frame is going slow-path. Effectively this

communication flow.	means that the frame will not be processed against an on-INIC CCB. It will be passed directly to the host, although if the frame is TCPIP, then its checksums have already been validated by the hardware. Also, all other header validations have been performed. If bit 29 is not set, then there may be an onboard CCB for this frame.”
62. The apparatus of claim 58, further comprising: a packet batching module configured to determine whether said packet memory includes another packet in said first communication flow.	 (86:1-8) “The ***receive sequencer has already generated*** a hash based on the network and transport addresses, e.g., ***IP source and destination addresses and TCP ports***. This hash is used to index directly into a hash table on the INIC that points to entries in a CCB header table. The header table entries are chained on the hash table entry. The microcode uses the hash to determine if a CCB exists on the INIC for this frame. It does this by following this chain from the hash table entry, and for each chained header table entry, ***comparing its source and destination addresses and ports with those of the frame.***”

Claims 24-26, 29-30, 41, 46, and 57:

Replicated below are copies of the claims in the present application that are not copied from U.S. Patent No. 6,427,169 (Claims 24-26, 29-30, 41, 46, and 57).

24. A method of transferring a packet to a computer system, wherein the packet is received at a communication device from a network, comprising:

parsing a header portion of a first packet received at a communication device to determine if said first packet conforms to a pre-selected protocol;

generating a transmit control block (TCB) to identify a first communication flow that includes said first packet;

transferring said first packet to a host computer system for processing in accordance with said pre-selected protocol; and

associating a summary with said first packet, wherein said summary indicates a status of said first packet.

25. The method of claim 24, further comprising:

transferring said TCB from said host computer system to said communication device.

26. The method of claim 24, further comprising:

receiving a second packet from a second communication flow; and

processing said second packet by said communication device in accordance with said TCB.

29. The method of claim 1, wherein said parsing includes determining, by hardware of the communication device, a session layer protocol of the header.

30. The method of claim 1, wherein said generating a flow key includes initializing a communication control block (CCB) during Transport Control Protocol (TCP) connection setup.

41. The method of claim 1, wherein said transferring said first packet to a host computer system comprises:

analyzing a session layer part of said header to allocate a list of memory addresses of the host computer for storing data from a plurality of packets that correspond to said communication flow.

46. The method of claim 1, wherein said operation code indicates whether the packet corresponds to Transport Control Protocol (TCP).

57. The method of claim 1, further comprising:

receiving, by said communication device, a second packet that corresponds to said communication flow, said second packet including at least a transport layer header and data; and

transferring said data from said communication device to said host computer system, without transferring said transport layer header from said communication device to said host computer system.

CONCLUSION

In view of the above remarks, Applicants request that an interference be declared between the present application and U.S. Patent No. 6,453,360. If the Examiner would like to discuss any aspect of this application, including how the claims are supported by Applicants' specification, the Examiner is requested to call the undersigned.

Respectfully submitted,

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Label No. EL928365371US in an envelope addressed to MS Patent Application, Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450, on June 19, 2003.

Date:______________ _________________
Mark Lauer

____
Mark Lauer
Reg. No. 36,578
7041 Koll Center Plaza,
Suite 280
Pleasanton, CA 94566
Tel: (925) 484-9295
Fax: (925) 484-9291

57. The method of claim 1, further comprising:

receiving, by said communication device, a second packet that corresponds to said communication flow, said second packet including at least a transport layer header and data; and

transferring said data from said communication device to said host computer system, without transferring said transport layer header from said communication device to said host computer system.

CONCLUSION

In view of the above remarks, Applicants request that an interference be declared between the present application and U.S. Patent No. 6,453,360. If the Examiner would like to discuss any aspect of this application, including how the claims are supported by Applicants' specification, the Examiner is requested to call the undersigned.

Respectfully submitted,

Mark Lauer
Reg. No. 36,578
7041 Koll Center Plaza,
Suite 280
Pleasanton, CA 94566
Tel: (925) 484-9295
Fax: (925) 484-9291

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as Express Mail Label No. EL928365460US in an envelope addressed to MS Patent Application, Commissioner for Patents, P.O. Box 1450 Alexandria, VA 22313-1450, on June 19, 2003.

Date: 6-19-03

Mark Lauer